CS 475/575 -- Spring Quarter 2024

Project #6

OpenCL Linear Regression

100 Points

Due: June 2

This page was last updated: May 22, 2024

Note:

- **The flip machines do not have GPU cards in them, so OpenCL will not run there.**
- ***rabbit* has a GPU in it, so you can run there. *rabbit* is especially good for developing and debugging.**
- **You can also use the DGX machine. The DGX is especially good for getting your final numbers.**
- **If your own system has a GPU with OpenCL support, feel free to use that.**

Introduction

These days there is a mad rush to be able to analyze massive data sets. In this project, you will perform a linear regression on a 4M (x,y) pair dataset. The linear regression will produce the equation of the line that best fits the data points. The line will be of the form: y = Mx + B (just like in high school...). Your job will be to run multiple tests to characterize the performance, and to determine what the values of M and B are.

Some Help

Here is proj06.cpp, a skeleton C++ program.
Here is proj06.cl, a skeleton OpenCL kernel program.
As web browsers don't know what the .cl file type is, they will try to force you to save proj06.cl
If you just want to see what is in this file right now, click here.

Requirements:

- You are to read the DATASIZE x-y pairs in the data file and store them in two host (CPU) arrays. The file is here: p6.data Left-click to look at it, right-click to download it.
  To be sure you have the entire file, the size of the file should be **79691776** bytes. The last 10 lines should be:

```
 0.73    11.34
 0.09     8.08
-0.85     3.03
-0.90     2.42
 0.41     8.08
-0.40     3.40
 0.36    10.10
-0.71     4.80
-0.58     4.15
 0.45    10.46
```

- You will then create two DATASIZE-sized device (GPU) arrays and copy all the (x,y) pairs into them.
- To perform a linear regression, one needs to produce 4 summations:
  1. Sum of all $x^2 = \Sigma x^2$
  2. Sum of all x = Σx
  3. Sum of all (x*y) = Σ(x*y)
  4. Sum of all y = Σy

  Trust me on this for now. I'd be happy to go over the derivation of why we need these sometime if you'd like.
- Create DATASIZE-sized host and device arrays for all four of these.
- In the proj06.cl file, compute these quantities for each (x,y) indexed with the OpenCL variable *gid*.
- Use different combinations of DATASIZE and LOCALSIZE and measure the performances.
- Use performance units that make sense. Joe Parallel used "MegaXYsProcessedPerSecond".

- Bring all four of these quantities back and place them in the previously-created host arrays.
- Add up the values in the host arrays to produce these CPU floats:
  1. Σx^2
  2. Σx
  3. $\Sigma(x*y)$
  4. Σy

  You can call these variables whatever makes sense to you.
- Use the Solve( ) function to turn these four quantities into the M and B line parameters (see below).
- Make two graphs:
  1. Performance versus DATASIZE, with a series of colored Constant-Local-Size curves
  2. Performance versus Local Size (LOCALSIZE), with a series of colored Constant-DATASIZE curves
- Your commentary PDF should tell:
  1. What machine you ran this on
  2. Show the table and graphs
  3. What patterns are you seeing in the performance curves? What difference does the size of data make? What difference does the size of each work-group make?
  4. Why do you think the patterns look this way?

Determining the M and B for the Line Equation

To complete the linear regression, you need to determine the optimal values of M and B from the line equation: y = M*x + B. To do this, you need to solve a two-equations-two-unknowns linear system. Fortunately, you don't need to figure out how to do this yourself. Call the Solve( ) function (it's in the skeleton code) like this:

```
float m, b;
Solve( Σx2, Σx, Σx, (float)DATASIZE, Σ(x*y), Σy,   &m, &b );
```

For those new to C/C++, the ampersand (&) means "address of". It gives a function a way to fill values that you want returned. When this function is done executing, **m** and **b** will have the computed values in them, ready for you to include in your report.

Developing this Project in Linux

You will need the following files:

1. cl.h
2. cl_platform.h

If you are on *rabbit*, compile, link, and run like this:

```
g++  -o  proj06  proj06.cpp  /usr/local/apps/cuda/10.1/lib64/libOpenCL.so.1  -lm  -fopenmp
./proj06
```

If you are on the *DGX System*, put these lines in a bash file:

```
#!/bin/bash
#SBATCH  -J  Proj06
#SBATCH  -A  cs475-575
#SBATCH  -p  classgputest
#SBATCH  --constraint=v100
#SBATCH  --gres=gpu:1
#SBATCH  -o  proj06.out
#SBATCH  -e  proj06.err
#SBATCH  --mail-type=BEGIN,END,FAIL
#SBATCH  --mail-user=joeparallel@oregonstate.edu

for s in 4096 16384 65536 262144 1048576 4194304
do
        for b in 8 32 64 128 256
        do
                g++ -DDATASIZE=$s -DLOCALSIZE=$b -o proj06 proj06.cpp /usr/local/apps/cuda/11.7/lib64/libOpenCL.so.1  -lm -fopenmp
                ./proj06
        done
done
```

and then submit that file using the *sbatch* slurm command.

Developing this Project in Visual Studio

Right-click on this link: VS2022.zip, and save it somewhere. Then, un-zip it, go to the VS2022 folder, and double-click on the **VS06.sln** file.

Getting the right platform and device number:

OpenCL is capable of running on multiple parts of the system you are on (CPU, GPUs, etc.). So, you must decide which one to use.

The skeleton code contains a function called SelectOpenclDevice( ) which selects what it thinks is the best Platform and Device to run your OpenCL code on. Call it from your main program. (That call is already in the skeleton code.) It will print out what it has decided to do. Feel free to comment out those print statements later so they don't interfere with producing an output file.

On *rabbit*, it produced:
I have selected Platform #0, Device #0: Vendor = NVIDIA, Type = CL_DEVICE_TYPE_GPU

Grading:

Feature	Points
Performance table	10
Graph of Performance versus DATASIZE	25
Graph of Performance versus LOCALSIZE work-group size	25
Determined the correct M and B values	15
Commentary	25
Potential Total	**100**

Note: the graph of performance versus DATASIZE needs to have colored curves of constant LOCALSIZE
Note: the graph of performance versus LOCALSIZE size needs to have colored curves of constant DATASIZE